IRIS: Integrate. Relate. Infer. Share.

Adam Cheyer, Jack Park, Richard Giuli

SRI International
333 Ravenswood Ave
Menlo Park, CA 94025
`<FirstName>.<LastName>@sri.com`

Abstract. In this paper we introduce a new semantic desktop system called IRIS, an application framework for enabling users to create a "personal map" across their office-related information objects. Built as part of the CALO Cognitive Assistant project, IRIS represents a step in our quest to construct the kinds of tools that will significantly augment the user's ability to perform knowledge work. This paper explains our design decisions, progress, and shortcomings. The IRIS project has grown from the past work of others and offers opportunities to augment and otherwise collaborate with other current and future semantic desktop projects. This paper marks our entry into the ongoing conversation about semantic desktops, intelligent knowledge management, and systems for augmenting the performance of human teams.

1 Introduction

In his 1962 paper, "Augmenting Human Intellect" [8], Douglas Engelbart wrote:

> *By "augmenting human intellect" we mean increasing the capability of a man to approach a complex problem situation, to gain comprehension to suit his particular needs, and to derive solutions to problems. Increased capability in this respect is taken to mean a mixture of the following: more-rapid comprehension, better comprehension, the possibility of gaining a useful degree of comprehension in a situation that previously was too complex, speedier solutions, better solutions, and the possibility of finding solutions to problems that before seemed insoluble. And by "complex situations" we include the professional problems of diplomats, executives, social scientists, life scientists, physical scientists, attorneys, designers--whether the problem situation exists for twenty minutes or twenty years. We do not speak of isolated clever tricks that help in particular situations. We refer to a way of life in an integrated domain where hunches, cut-and-try, intangibles, and the human "feel for a situation" usefully co-exist with powerful concepts, streamlined terminology and notation, sophisticated methods, and high-powered electronic aids.*

In 2005, Engelbart's statement remains both prescient and compelling. That 1962 paper correctly predicts a networked "global village," as Marshall McCluhan [13] would later call it. We now live and work in that global village, and the term *infoglut* has become the meme that reminds us of the information overload we experience in our daily lives, and about which Vannevar Bush so eloquently wrote in his 1945 paper, "As We May Think" [2]. Indeed, it was that paper which inspired Ted Nelson, Douglas Engelbart, and many others to try to find solutions to the infoglut problem and augment human capabilities for solving complex, urgent problems.

Charles Bourne and Douglas Engelbart open their 1958 paper, "Facets of the Technical Information Problem," [1] with:

> *RECENT world events have catapulted the problem of the presently unmanageable mass of technical information from one that should be solved to one that must be solved. The question is receiving serious and thoughtful consideration in many places in government, industry, and in the scientific and technical community.*

This is a long version of a talk presented at the First Semantic Desktop Workshop

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE **2005**	2. REPORT TYPE	3. DATES COVERED **00-00-2005 to 00-00-2005**
4. TITLE AND SUBTITLE **IRIS: Integrate, Relate. Infer. Share.**		5a. CONTRACT NUMBER
		5b. GRANT NUMBER
		5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)		5d. PROJECT NUMBER
		5e. TASK NUMBER
		5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **Artificial Intelligence Center,SRI International,333 Ravenswood Avenue,Menlo Park,CA,94025**		8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)		10. SPONSOR/MONITOR'S ACRONYM(S)
		11. SPONSOR/MONITOR'S REPORT NUMBER(S)
12. DISTRIBUTION/AVAILABILITY STATEMENT **Approved for public release; distribution unlimited**		
13. SUPPLEMENTARY NOTES **The original document contains color images.**		
14. ABSTRACT		
15. SUBJECT TERMS		

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			
unclassified	**unclassified**	**unclassified**		**13**	

If networked computers will be the "printing presses of the twenty-first century" and beyond, then networked *semantic desktop* applications will be the workstations of many of those knowledge workers mentioned by Stefan Decker and Martin Frank in their 2004 paper, "The Social Semantic Desktop" [6]. Today, knowledge workers are accustomed to the use of applications such as email, calendar, word processing, spreadsheets, and more. Each of those applications can be viewed as stand-alone entities, each facilitating the accomplishment of some particular task, but in no particular sense *integrated* in ways we shall call *semantic* with each other. An appropriate interpretation of John Stuart Mill's 1873 [14] call for fundamental changes in our modes of thought would suggest that we look at *semantic integration* of the tools with which we perform knowledge work.

In this paper, we will introduce a new semantic desktop system called IRIS (for Integrate-Relate-Infer-Share) and explain the context in which it was built, as part of the CALO Cognitive Assistant project. As we describe our quest to construct the kinds of tools that will significantly enhance the desktop user's experience and augment the user's ability to perform knowledge work, we will explain our design decisions, progress, and shortcomings. With IRIS, we have attempted to grow our project out of the past work of others, and, we believe, offer opportunities to augment and otherwise collaborate with other current and future semantic desktop projects. In this paper we take part in the ongoing conversation about semantic desktops, intelligent knowledge management, and systems for augmenting the performance of human teams.

2 Background and Requirements

> *I believe that at the end of the century the use of words and general educated opinion will have altered so much that one will be able to speak of machines thinking without expecting to be contradicted.*
>
> –Alan Turing, *Computing Machinery and Intelligence* 1950

Before discussing the IRIS project, we will briefly describe CALO[1], an artificial intelligence application for which IRIS serves as the semantic desktop user interface. Requirements from the CALO program have greatly influenced our design for IRIS.

IRIS has been developed as part of SRI's CALO[2] project, one of two projects funded under DARPA's "Perceptive Assistant that Learns" (PAL) program[3]. The goal of the PAL program is to develop an enduring personal assistant that "learns in the wild," evolving its abilities more and more through automated machine learning techniques rather than through code changes. DARPA expects the program to generate innovative ideas that result in new science, new and fundamental approaches to current problems, and new algorithms and tools, and to yield new technology of significant value to the military and commercial sectors. Led by SRI International, 250 researchers and developers from 25 universities and companies are working on CALO.

CALO is a cognitive software system that can reason, learn from experience, be told what to do, explain what it is doing, reflect on its experience, and respond robustly to surprise. CALO's mission is to serve its user as a personal assistant, collaborating in all aspects of work life: organizing information; preparing information artifacts; mediating person-person interactions; organizing and scheduling in time; monitoring and managing tasks; and acquiring and allocating resources.

To understand and reason about the dynamics of the user's work life, CALO requires a semantically coherent view into the user's life and a mechanism for interacting with the user in a natural work setting. Our solution was to outfit CALO researchers with a semantic desktop, called IRIS, that enables them to outline the key elements in their environment, specifically: the projects the user works on; the key participants in various roles for these projects; the way in which accessed information relate to people, projects, and tasks in the user's life; and the priorities of tasks, messages, documents, and meetings. CALO plays the role of a collaborative teammate participating in this exercise, learning how to populate much of the semantic content and relationships on behalf of the user and the rest of his team.

In approaching the design and development of IRIS, we took much inspiration from the work of Douglas Engelbart. While Ted Nelson's Xanadu[4] [15] was arguably the first project to set the stage for modern hyper-

[1] CALO: http://www.ai.sri.com/software/CALO

[2] CALO is an acronym for "Cognitive Assistant that Learns and Organizes." CALO's name was also inspired by the Latin word *calonis*, which means "soldier's servant" and conjures an image of Radar O'Reilly from the M*A*S*H TV series.

[3] DARPA's PAL program: http://www.darpa.mil/ipto/programs/pal/

[4] Xanadu: http://xanadu.com/

document processors, Engelbart's Augment[5] was the first system to actually find engagement in group document processing and sharing. In 1968, at the Fall Joint Computer Conference in San Francisco, Engelbart demonstrated Augment before a live audience.[6] Augment displayed many of the capabilities we now want to build into modern semantic desktop applications. Augment, the program, saw commercial application, and is still used today by Dr. Engelbart in his day-to-day activities. Efforts are under way to create open source variants of the Augment system [19]. At the same time, work continues on the development of an Open Hyperdocument System [9] guided by Dr. Engelbart.

CALO, Engelbart's work, the paper by Decker and Frank [6], and an earlier paper by Gradman [10] combine to provoke background thoughts that drive the evolution of our requirements. Here are several that animate the IRIS project:

1. *"Real" enough to do daily work.* As dictated by Engelbart's notion of bootstrapping,[7] we should develop using that which we are developing. To convince people to give up their current mail program, web browser, or calendar in favor of IRIS, we will need to offer a full-featured experience that supports all of their specific needs: mail encryption, spam filters, calendar servers, synchronization with PIMs[8], embedded flash, etc. Rather than implement these features ourselves, we opt to find and integrate the most mature third-party applications available into our developments.
2. *Implemented in, or able to easily integrate with, Java*. This requirement comes from the fact that many of the machine learning components we will include from CALO researchers are implemented in Java.
3. *Ontology-based knowledge store*. We require the ability to model rich semantic structures that can capture every aspect of a user's work environment. The semantic representation should be able to interoperate with KM[9] [4], a full first-order logic reasoning engine that is used as the heart of CALO's "thinking." We want to minimize the differences between CALO's internal representation and the way data and events are represented within IRIS.
4. *Capable of supporting organization of personal knowledge assets*. This implies providing for the ability of users to organize their information resources in ways that suit individual needs ("just for me") while maintaining semantic interoperability with other CALO installations.
5. *Cross-platform*. IRIS should be able to run on Windows, Macintosh, and Linux platforms, to support as widely as possible the diverse CALO community.

3 Related Work

With these requirements in mind, we set about looking at candidate solutions that could meet our needs. We started by looking at Mitch Kapor's Chandler[10] project, which certainly belongs in the semantic desktop category. The Chandler web site says this:

> *With Chandler, users will be able to organize diverse kinds of information for their own convenience -- not the computer's convenience. Chandler will have a rich ability not only to associate and interconnect items, but also to gather and collect related items in a single place creating a context sensitive "view" of many types of data, mixing-and-matching email, mailing lists, instant messages, appointments, contacts, tasks, free-form notes, blogs, web pages, documents, spreadsheets, slide shows, bookmarks, photos, MP3's, and so on. Data in Chandler is stored on repositories on the user's local machine, on others' machines, and on shared resources such as servers.*

While Chandler's vision resonated with what we wanted for IRIS, its early stage of development and long product roadmap made Chandler an unsuitable starting point.

We next explored Haystack[11] from MIT. When we discovered this project [12], we were amazed how well it fit our initial designs for IRIS, in terms of both architecture and user interface design, with the added benefit of

[5] NLS/Augment at the Computer History Museum: http://community.computerhistory.org/scc/projects/nlsproject/
[6] Videos of the first online document editing project. Found on the web at http://sloan.stanford.edu/MouseSite/1968Demo.html
[7] Engelbart's work on bootstrapping productivity: http://www.bootstrap.org
[8] Personal information managers
[9] KM: http://www.cs.utexas.edu/users/mfkb/RKF/km.html
[10] Chandler: http://www.osafoundation.org/

being Java-based and open source. We invited Dennis Quan to visit SRI to discuss the internals of Haystack in relation to our perceived needs. We learned much from the visit and did, indeed, begin the task of adapting Haystack's significant code base to our framework. For a variety of reasons, we ended up moving in a different direction, but Haystack and Dr. Quan's deep knowledge of the subject gave us a solid start.

The next system we evaluated was the Radar Networks[12] Personal Radar, a very impressive semantic desktop that turned out to share many of the goals and requirements for IRIS: Java-based, ontology-driven, user centric. We negotiated, and CEO Nova Spivack agreed to join the CALO project to help combine elements of Personal Radar into the IRIS code-base. In particular, we adopted their Semantic Object framework, a very fast triple-store implementation, and certain elements from their SWING-based user interface. These are described in more detail in Section 4.

Well down the path of implementing IRIS, we came across Gnowsis[13]. Gnowsis [18] appears to offer integration with many of the same third-party applications as IRIS, and to share many similar philosophies regarding application and data integration. Where IRIS and Gnowsis currently diverge may lie in the way in which those applications are integrated. Whereas Gnowsis appears to have fairly loose integration with standalone applications, using adapters to copy references from applications into a local server and a separate browser for navigating and searching the data, IRIS has chosen to be more tightly integrated at the user interface level, providing an "embedded suite" of applications. Each plug-in application is instrumented such that IRIS captures semantic events as they occur. For example, IRIS "knows" which web page is being browsed, or which email has been opened for reading. Tight integration of applications is particularly useful to IRIS's learning framework and components, which can offer real-time suggestions as the user works with information.

Most recently, we discovered MindRaider[14], a project arguably close to IRIS, Haystack, and Gnowsis in spirit and intent. MindRaider's open source license precludes us from looking closely at the source code, but we observe, while running the program, that there are profound similarities between its ontology-driven architecture and that of IRIS, and we note several interesting parallels in user interface design. We suspect that HCI (user interface design) will eventually rise to be at least as important to the success of semantic desktops as is semantic interoperability among platforms.

Topic Maps [17] is another research area we are tracking, as we consider IRIS a kind of topic map for personal information assets. A topic map[15] provides a container for proxies for subjects, called topics. Each subject, which is anything that can be the focus of thought or discussion, is represented by one topic. Each topic is a kind of container for links to all known information about the subject. Topics have properties that provide subject identity and other properties of the subject. Topics can play roles in association with other topics. For instance, a topic, which is a proxy for the IRIS user, can play roles such as member in a meeting, speaker at a conference, parent or spouse, and more. Topics are associated with occurrences. For instance, the topic for a particular personal computer would be linked with occurrences such as web pages where that computer can be purchased, or where an online help system is found. Topics are also repositories for all possible ways to *name* the topic. With this, individuals can assign names for things in their personal space; that personal space thus gains the "just for me" flavor. Subject identity does not change when a different name is given to a topic; semantic interoperability demands uniform, consensus subject identity, and a topic map maintains that.

4 The IRIS Semantic Desktop

IRIS is an application framework for enabling users to create a "personal map" across their office-related information objects. IRIS is an acronym for "Integrate. Relate. Infer. Share." In the following text, we will adopt these four terms as organizing subsections, as we describe IRIS's design, architecture, implementation, and use-cases.

---

[11] Haystack: http://haystack.lcs.mit.edu/
[12] RadarNetworks: http://www.radarnetworks.com/
[13] Gnowsis: http://www.gnowsis.org/
[14] MindRaider: http://mindraider.sourceforge.net/
[15] Topic maps: http://www.topicmaps.org/

3.1 IRIS – Integrate

Hypertext is a form of storage, a new form of literature, and a network that just might revitalize human life.

–Ted Nelson 1965

IRIS is first and foremost an integration framework. Whereas in today's packaged applications suites, where only loose data integration exists[16] (usually limited to the clipboard and common look-and-feel for menus and dialog boxes), IRIS strives to integrate data from disparate applications using reified semantic classes and typed relations. For instance, it should be possible to express that "*File F* was presented at *Meeting M* by *Tom Jones,* who is the *Project Manager* of *Project X*," even if the file manager, calendar program, contact database, and project management software are separately developed third-party applications. In a Topic Maps fashion, there should be a single instance that represents each concept, and all that is knowable about that concept should be directly accessible from that instance [17].

The IRIS framework offers integration services at three levels (Figure 1):

1. Information resources (e.g., an email message, a calendar appointment) and the applications that create and manipulate them must be made accessible to IRIS for instrumentation, automation, and query. IRIS offers a plug-in framework in the style of the Eclipse[17] architecture or the JPF framework[18], where "applications" (components with a graphical user interface) and "services" (processing components with no GUI of their own) can be defined and integrated within IRIS. Apart from a very small, lightweight kernel, all functionality within IRIS is defined using the plug-in framework, including user interface, applications, back-end persistence store, learning modules, harvesters, and so forth.

2. A knowledge base (KB) provides the unified data model, persistence store, and query mechanisms across the information resources and semantic relations among them. Ontology services are described in more detail in Section 4.2, Relate.

3. The IRIS user interface framework allows plug-in applications to embed their own interfaces within IRIS, and to interoperate with global UI services such as the notification pane, menu and toolbar management, query interfaces, the link manager, and suggestion pane.

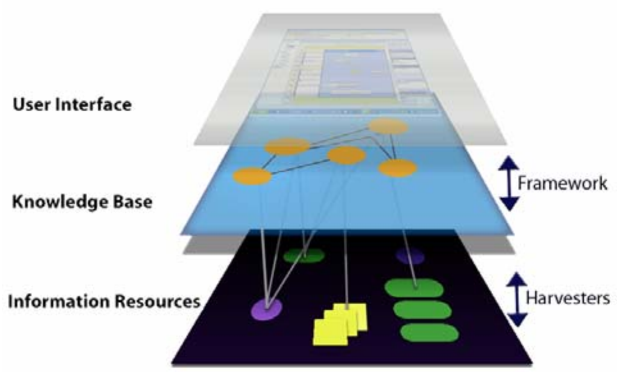

Figure 1: The three-layer IRIS integration framework.

IRIS comes "out of the box" with several integrated office applications:

- *Email*: After initially integrating Java-based Columba,[19] we moved to Mozilla[20] for email, as it is one of the most popular, full-featured, cross-platform applications available. To integrate

[16] Even within a single application, deep data integration is usually pretty threadbare. Consider Microsoft Outlook: the email addresses displayed in a message are not linkable (or deeply related) to the people records in your contacts folder.

[17] Eclipse: http://www.eclipse.org/

[18] Java Plugin Framework (JPF) Project: http://jpf.sourceforge.net/

[19] Columba Mail: http://columba.sourceforge.net/

[20] Mozilla Application Suite: http://www.mozilla.org

Mozilla with Java, we adopted and made significant extensions to the JREX[21] package, and then ran Email as an embedded XUL[22] application.

- *Web browser*: Mozilla provides an infinitely better web browsing experience than our initial integration effort, Java-based CALPA.[23]
- *Calendar*: We selected OpenOffice GLOW[24] because it is Java-based and iCAL compliant, interoperates with the Sun/Netscape calendar server used by SRI, and has a very nice user interface. Mozilla's calendar is very primitive at this stage. However, GLOW is somewhat buggy and the code is quite messy, so we are in the market for a replacement.
- *Chat*: We implemented our own interface to the Jabber[25] instant messaging backend.
- *File explorer*: We wrote our own in Java.
- *Data editor/viewer*: To view and edit data records such as people, projects, tasks, and any other ontology-based object in the KB, we used a forms package from Radar Networks' Personal Radar software.

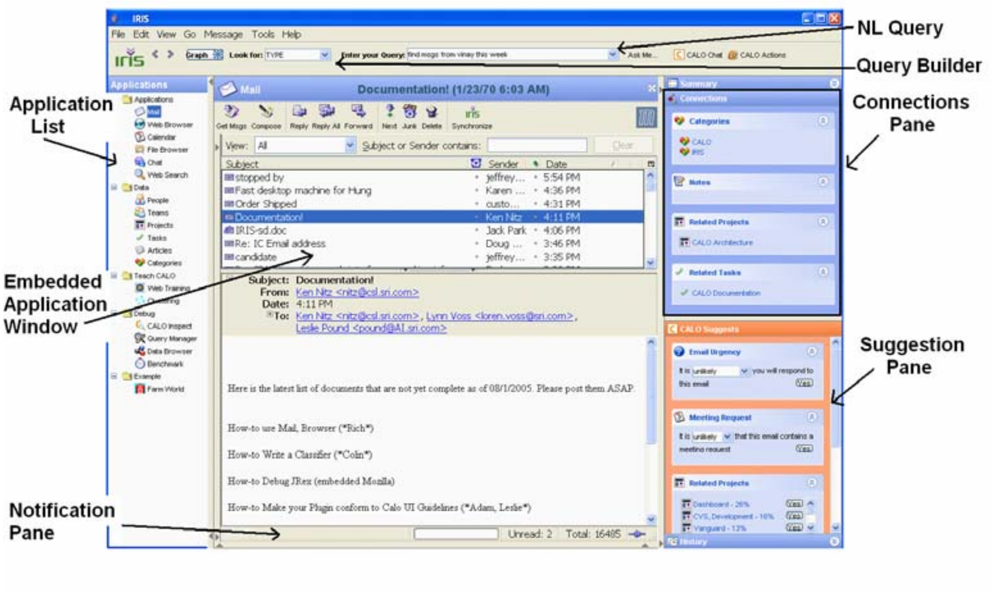

Figure 2: The IRIS user interface.

The IRIS user interface provides the "shell" for hosting each of these embedded applications (figure 2). Two side panels frame the main application window, one for selecting among available applications, the other for displaying and editing semantic links for the selected application object and presenting contextual suggestions from the learning framework. Applications can add toolbars to the IRIS frame, and when selected, an application's menu items are "merged" with IRIS menu functionality present for all applications. IRIS provides an extensible context-sensitive online help system and several methods for querying information resources within and across applications. An example of a natural language query supported by iAnywhere's Answers Anywhere[26] IRIS plugin is "find email from Vinay last week related to the CALO project."

[21] JREX – Mozilla through Java: http://jrex.mozdev.org/
[22] XUL – http://www.xulplanet.com/
[23] CALPA: http://htmlbrowser.sourceforge.net/
[24] OpenOffice GLOW calendar: http://groupware.openoffice.org/glow/
[25] Jabber Instant Messaging: http://www.jabber.org/
[26] Answers Anywhere NL query: "http://www.ianywhere.com/products/answers_anywhere.html

4.2 IRIS – Relate

> *Information is both more and less real than the material universe. It's more real because it will survive any physical change; it will outlast any physical manifestation of itself. It's less real because it's ineffable. For example, you can touch a shoe, but you can't touch the notion of "shoe-ness" (that is, what it means to be a shoe). The notion of shoe-ness is probably eternal, but every shoe is ephemeral.*
>
> – Steven R. Newcomb [17, page 32]

IRIS is used to *semantically integrate* the tools of knowledge work. What do we mean by this? We use the term "semantic" in the sense used by the Semantic Web community, where markup technologies are being wedded to the tools of semantic representation (e.g., ontologies, OWL, RDF). This facilitates putting data on the web in such a way that machines can access it, make meaningful references to it, and perform manipulations on it, including reasoning and inference. In that sense, IRIS provides a knowledge representation by which the artifacts of a user's experience such as email messages, calendar events, files on the disk or found on the web, can be stored and related to each other across applications and across users.

When defining the ontology to be used for IRIS, a design choice had to be made: Do we use a small, simple ontology or a complex, more-expressive ontology? We first implemented a fairly large, yet straightforward, ontology. However, the requirement that IRIS interoperate with CALO's reasoning and learning capabilities drove us to adopt CALO's pre-existing ontology, which supports roles, events, and complex data structures.

CALO's ontology is called CLib,[27] the Component Library Specification, which consists of definitions for everyday objects and events, as well as axioms to support the beginnings of common-sense reasoning. CLib is implemented in KM (for "The Knowledge Machine"), a frame-based language with first-order logic semantics and a reasoning engine written in Lisp. Note that KM is not used as the knowledge repository for IRIS because it is not designed for manipulating the hundreds of thousands of data instances a user-centric desktop application requires. However, it is the goal of IRIS to represent data and events equivalent to KM language constructs. To accomplish this goal, a translator has been written to convert the KM definitions (just the data definitions, not the axioms) from CLib to OWL. We chose OWL[28] as the data representation in IRIS because it is a W3C-approved standard that allows for a flexible data schema. Currently IRIS supports the OWL Lite subset, with future plans to support OWL DL.

The IRIS development framework provides services to simplify access and improve maintenance for interoperating with the OWL-based data structures from Java. At compile time the IRIS SDK generates "POJOs" for each class in the ontology. POJO stands for "Plain Old Java Object" or "Programmable Ontology Java Object." POJOs provide references to other POJOs in property relations to build a complete API for ontology access. For example, an IEmailMessagePojo contains methods to access the header (IEmailMessageHeaderPojo), body (IMessageBodyPojo), addresses (IEmailAddressPojo), and so on. Methods generated on the POJO correspond to direct properties on the ontology class, and also include indirect ontology paths to other data known as *binding paths*.

Binding Paths, are another way IRIS attempts to reduce the overhead of accessing a complex ontology. They are used to describe common ontology traversals. The IRIS ontology contains definitions for BindingPathProperty instances that describe an ontology traversal and restrictions on the traversal. A BindingPathProperty is also an ontology property so that it can be accessed as a top-level property on a class even though accessing the property requires traversals of more than one relation and instance. Furthermore, in the cases that the traversal passes through anonymous instances, rules can be specified for the creation of the anonymous classes. Binding Paths are interpreted at the lower ontology level, and support is also provided for querying using a BindingPathProperty. Note that binding paths are provided purely for simplicity in accessing the ontology from a software standpoint; binding path URIs never make their way into the database.

Binding Paths are also incorporated as methods on the POJOs to simplify the programming API. For example, a typical Binding Path might be getting/setting the work email of a person, which requires stepping through

---

[27] KM Component Library: http://www.cs.utexas.edu/users/mfkb/RKF/tree/ then select "Core+Office" to browse the CALO subset.

[28] Web Ontology Language: http://www.w3.org/TR/owl-features/

several relations. When retrieving the IPersonPojo that is the sender of an IEmailMessagePojo, a complex ontology path must be traversed that walks through the header, address, contact information for the address, and the person associated with the contact information. All this work is encapsulated in the POJO implementation to simplify ontology access while keeping the intended structure intact. A software engineer simply calls the SetWorkEmail() method on a Person object.

In between the POJO (highest) layer and lowest (implementation) layer of ontology access is an abstraction layer that hides the ontology implementation. This API for ontology access is known as the Semantic Object Framework (SOM). The SOM abstracts ontology instances, properties, and classes as Java objects with a programming API for ontology access. POJOs are built on top of the SOM. Note that POJOs differ from the SOM in that POJOs classes are generated for each ontology class providing an *ontology API,* whereas the SOM is an API for accessing an abstract ontology. POJOs are important because they keep the code consistent at compile time, and when the ontology is changed, programming errors are quickly found. The SOM is provided when POJOs do not suffice, but in general POJOs are the preferred means of accessing the ontology. One advantage of having the SOM layer is that backend implementations can be exchanged in a plug-and-play fashion.

IRIS incorporates the use of a formal Documentation Ontology. This ontology provides classes and methods used to document the ontology usage for an IRIS application. For a POJO to be generated, there must first be documentation written in RDF for the class and method(s) to be used, and each application in IRIS that uses the ontology is expected to produce an RDF document for the ontology usage. Each piece of documentation written contains a reference to the ontology class, one or more references to the properties used for that class, and HTML documentation for all references. The advantage of writing this usage documentation using an ontology is that the documentation refers directly to ontology classes and properties in RDF, which allows the documentation to be interpreted semantically.[29] It is required that documentation be formally written in this manner to appropriately document and assemble ontology usage in IRIS. This documentation exists as a set of RDF files and can be viewed using an XSL transformation. The use of this documentation ontology in auto-generating POJOs keeps the documentation intact with the code. Note that documentation for Binding Paths is also generated using the documentation ontology.

IRIS provides a framework for *harvesting* application data and *instrumenting* user actions in IRIS applications. The harvesting of data refers to importing external data into semantic (ontology-based) structures. For example, if given the specification of an email instance, harvesting APIs exist to create ontology structures for the email, addresses, and people associated with those addresses. These ontology structures are available in the instrumentation API for application events. This data is then translated once again to an external event publish/subscribe model that allows other IRIS plug-ins or external applications to access the data. Note that external instrumented events do not conform to an ontology structure because they are meant to be concise, small data structures. However, those data structures are generated from the more complex ontology structures that are created after harvesting. Finally, note that the documentation ontology is also used to document instrumented events.

Currently IRIS uses a custom ontology engine provided by Radar Networks as the SOM implementation. The IRIS team has recently created a SOM implementation built on JENA models, with a fast database implementation underneath. Furthermore, full text indexing has been added using a Lucene search engine. The switch to JENA is almost complete. Using JENA has some advantages: It comes with a standard set of reasoners, defines a flexible graph architecture, and allows support for RDQL[30] and SPARQL[31] queries.

34.3 IRIS – Infer

If you invent a breakthrough in artificial intelligence, so machines can learn, that is worth 10 Microsofts.

–Bill Gates, quoted by NY Times[32]

[29] Note that currently the IRIS SDK reads the ontology documentation to generate POJOs, and also uses the documentation to generate CALO-specific documents for integration with other CALO modules.

[30] RDF Data Query Language: http://www.w3.org/Submission/2004/SUBM-RDQL-20040109/

[31] SPARQL Protocol and RDF Query Language: http://www.w3.org/TR/rdf-sparql-query/

[32] Gates speech: http://www.nytimes.com/2004/03/01/technology/01bill.html

One of the key differentiators of IRIS, compared to many semantic desktop systems, is the emphasis on machine learning and the implementation of a plug-and-play learning framework. We see machine learning as one of the solutions around a key issue limiting the semantic web's growth and mass adoption: Who is going to enter all of the required links and knowledge?

Here we present a typical use case of how learning components integrated within the IRIS framework combine to progressively construct a semantic representation of the user's work life.

Step 1: Email Harvesting: As the user receives email in Mozilla, IRIS automatically harvests messages, adding them as semantic instances in the knowledge base. As part of this process, names in the address fields are normalized (e.g., "Rich Giuli" will match "Richard Giuli"), links are created to existing contact records in the KB, and new records are added for people not in the KB. Events indicating new email messages and people records are published to the Instrumentation Bus for other learning components to consume.

Step 2: Contact/Expertise Discovery: When contact records containing a name and email address are added to the KB, the DEX service (from UMass), a CALO component, wakes up and tries to discover additional information for that person by applying information retrieval techniques (Conditional Random Fields, see [5]) to web pages owned by that person at the domain specified by the email address. Contact information is discovered, as well as a "gist" representing a person's expertise, composed of keywords and noun phrases that are significant for the person (Figure 3).

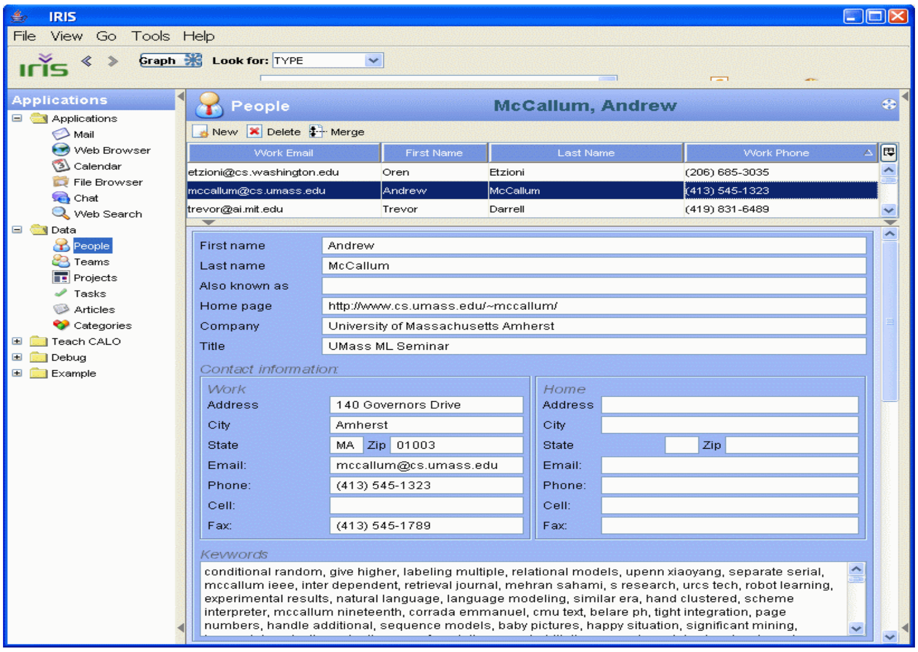

Figure 3: Information automatically extracted for people by harvesting the web.

Step 3: Learn from Files: In a similar fashion to email, IRIS harvests information from files on the user's desk. Currently, SEMEX [7] (from UWashington) opens LaTeX, BiB, and Microsoft Office files (Word, Excel, PowerPoint) to add content (e.g., publication references) to people in the contact KB.

Step 4: Project Creation: Clustering algorithms in IRIS are applied to the user's email to propose new projects to be added to the KB. For each project instance, a label for the project is proposed using the most salient phrase in the email cluster, keywords are added that provide a "gist" of the project, and links are added to project participants using the people in the from/to fields for the email cluster. IRIS provides a user interface where the differences between multiple clustering algorithms can be explored. Currently, three algorithms have been integrated into the framework: Carrot2/Lingo, based on singular value decomposition [16]; an algorithm based on agglomerative clustering and social network analysis [11]; and an algorithm based on linear optimization with user-specified centroids.

Step 5: Classification According to Project: Leveraging the textual content and relations extracted for projects, people, and files, a Bayesian classifier is applied to hypothesize relationships between projects and objects such as emails, files, web pages, and calendar appointments. IRIS's suggestions are displayed to the user, who can optionally provide feedback to the algorithm by indicating the correct values (Figure 4).

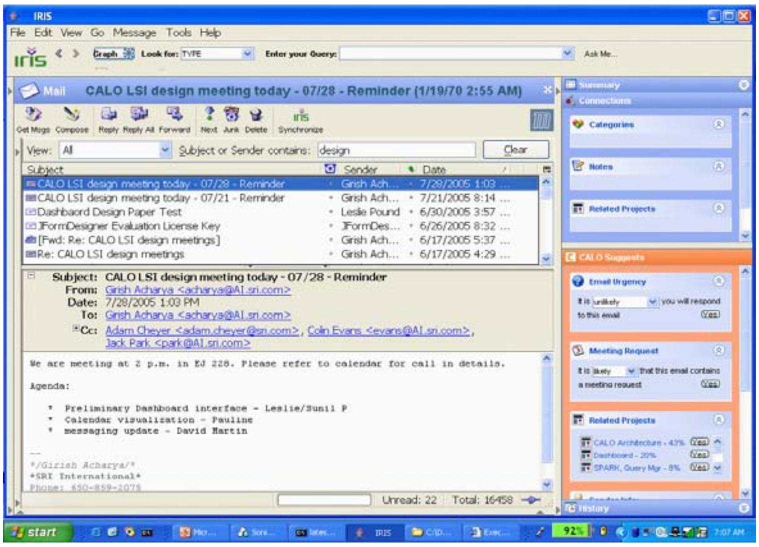

Figure 4: In the "CALO Suggests" pane, learned hypotheses about an email are presented, including reply urgency, meeting detection, project association, and others. The user can provide feedback about the system's choices, and the system will adapt accordingly.

Step 6: Higher-level Reasoning: A number of specialized reasoners within CALO continually examine events in the user's activity stream and attempt to make useful inferences. For instance, when the user clicks on an email, IRIS attempts to predict whether the user will/should reply to the message (Figure 4). Another plug-in applies text summarization techniques to produce a gist that will be faster for the user to read. When a calendar appointment is selected, CALO assembles a "PrepPack" of resources (e.g., email attachments, web pages, files, other appointments) that could be useful when preparing for or attending the selected meeting. For email threads, a plug-in tracks the "speech-acts" present in the conversation, extracting classifications such as "REQUEST-MEETING" or "PROPOSE-DELIVERABLE" [3]. In several use cases, multiple reasoners are combined to produce a single prediction. If the user provides negative feedback to a resulting hypothesis, each individual reasoner will adapt itself accordingly, and a meta-learner will use the intermediate results from each predictor to improve its own logic about how their results are combined. Implemented examples of this approach include the "Meeting PrepPack" reasoner and the "Meeting Request" detector.

This use case gives readers a flavor of the types of learning components that have been integrated within IRIS to help construct and then leverage a semantic model representing the user's work life. We feel that we have just scratched the surface of the types of useful learning-based functionality that can be integrated into the IRIS semantic desktop, and we are eagerly anticipating continued development, working with members of the CALO and open source communities.

3.4 IRIS – Share

Prior to the Internet, the last technology that had any real effect on the way people sat down and talked together was the table.

–Clay Shirky, at Emerging Technology Conference 2003[33]

[33] Clay Shirky: http://shirky.com/writings/group_enemy.html

Sharing information is one of the four key concepts that make up the IRIS vision. We feel that the ability to learn and leverage semantic structure in organizing one's work life will be greatly enhanced in a collaborative setting. Shared structures are essential for both end-user applications, such as team decision making and project management, and for infrastructural components such as machine learning algorithms, which improve when given larger data sets to work on.

In the first version of IRIS, we experimented with a simple collaborative functionality using a Jabber-based transport mechanism. Changes to shared data were written to a "chat room" space representing an ACL group; each IRIS client would remember what changes it had seen previously, and upon startup, initialize its KB by applying all recently recorded change actions. This approach had the benefit that it supported real-time collaborative work between online participants as well as enabling "late-comers" to join and be initialized to the common state. However, with no locking mechanisms, users were plagued by inconsistencies, and we temporarily removed the collaboration feature.

In the coming year, the CALO project has planned functionality and applications that will require infrastructure to support collaborative team decision making, as well as reasoning over a shared document space. We are therefore currently revisiting what approaches to take regarding a collaboration infrastructure for IRIS.

5 Evaluations & Conclusions

IRIS is now in daily operation as the primary office environment used by several of the authors and a few other members of the CALO community. In addition, as part of the formal evaluations of the CALO project, IRIS and its learning components were used extensively by 15 users during a few weeks of testing, giving CALO an opportunity to learn "in the wild" through observation and interaction with its user. After this experimentation period, we interviewed the users to understand what they liked, what features were missing, and how IRIS generally should be improved.

We were encouraged that most of the feedback was quite positive, with many of the users stating that they were generally pleased with the robustness of the system and impressed with the capabilities of IRIS to learn and provide useful data to them. In particular, the capability to automatically discover contact and "gist" information for people from whom they receive email was much appreciated. Several reports of events where IRIS made a significant positive contribution were noted. For instance, one user, after skimming a long email from his boss, wondered why IRIS was flagging the message as a meeting request. Upon closer reading, he discovered that towards the end of the message, his boss was actually requesting a meeting with him later that afternoon, with an expected deliverable.

Despite the positive feedback, a number of issues still need to be addressed before most users will be willing to adopt IRIS as their primary work environment:

1. Performance was the Number 1 issue. Many noted that the startup time for IRIS was quite slow, and actual use was sluggish, particularly during email use, where many research-quality components would process each selected message. Subsequent analysis revealed that 68% of the startup time and 72% of memory use could be attributed to three learning components; these will be candidates for optimization in the future.

2. Many user interface issues were noted, in particular regarding real-estate management for smaller screens, several inconsistencies in UI design, and the desire to use drag-and-drop (IRIS currently uses pop-up choosers to define relational links between objects). Also, several minor user interface bugs were mentioned, the most annoying being a proclivity for IRIS to pop up or become the selected window in an unsolicited way when activity (such as the arrival of an important email message) occurs. Improvements to the user interface experience will become a significant priority for the near-term roadmap.

3. Finally, many users, excited by the glimmer of intelligence that IRIS (and the cognitive assistant CALO) at times seemed to exhibit, made numerous suggestions about use cases they thought the pair would be able to help them with in the future to aid their productivity.

In the coming months, we will aggressively pursue these and other roadmap items, with a particular emphasis on making IRIS usable and useful for collaborative teams. We remain enthusiastic about the potential for coupling semantic representations, machine learning, user interface design, and real-world office systems.

5 Acknowledgments

This work was supported by the Defense Advanced Research Projects Agency (DARPA) under Contract No. NBCHD030010. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of DARPA or the Department of Interior-National Business Center (DOI-NBC).

We would like to thank Nova Spivack and Jim Wissner at Radar Networks for their tremendous contributions to the code base, ontologies, and vision for IRIS. Many members of the CALO LSI team worked hard to make IRIS what it is: Colin Evans, Steve Hardt, Jim Carpenter, Ken Nitz, Ayse Onalan, Leslie Pound, Girish Acharya, Mark Gondek, Talia Shaham, Julie Wong, Ken Doran, David Dunkley, Chris Brigham, and Jason Rickwald. Thanks to the CALO management team for supporting the concept: Bill Mark, Ray Perrault, David Israel, Jim Arnold, and Jeffrey Davitz. Final thanks to those non-SRI CALO members, too many to name, who are working with us to integrate their cutting-edge technologies into the IRIS learning framework.

References

1. Bourne, Charles P., and Douglas C. Engelbart, "Facets of the Technical Information Problem," *SRI Journal*, Vol. 2, No. 1, 1958. On the web at http://bootstrap.org/augdocs/friedewald030402/facets1958/Facets1958.html
2. Bush, Vannevar, "As We May Think," The Atlantic Monthly, July, 1945. On the web at http://www.theatlantic.com/doc/194507/bush
3. Carvalho, Vitor and Cohen, William. "On the Collective Classification of Email Speech Acts." The 28th Annual International ACM SIGIR Conference, Salvador, Brazil. August 2005. On the web at http://www.cs.cmu.edu/~wcohen/postscript/sigir-2005.pdf
4. Clark, Peter, and Porter. "Building Concept Representations from Reusable Components." In AAAI'97, pages 369-376, CA. AAAI Press, 1997. On the web at http://www.cs.utexas.edu/users/pclark/papers/aaai97.pdf
5. Culotta, Aron; Bekkerman, Ron; McCallum, Andrew. "Extracting Social Networks and Contact Information from Email and the Web." In Proceedings of CEAS, First Conference on Email and Anti-Spam (CEAS). July 2004. On the web at http://www.cs.umass.edu/~culotta/pubs/ceas04.pdf
6. Decker, Stefan; and Martin Frank, "The Social Semantic Desktop," 2004. On the web at http://www.deri.at/publications/techpapers/documents/DERI-TR-2004-05-02.pdf
7. Dong, Xin; Halevy, Alon; Nemes, Ema; Sigurdsson, Stephan. "SEMEX: Toward On-the-fly Personal Information Integration." Workshop on Information Integration on the Web (IIWEB). Toronto, CA. August 2004. On the web at http://data.cs.washington.edu/papers/semex_iiweb.pdf
8. Engelbart, Douglas, "Augmenting Human Intellect," October, 1962. On the web at http://www.bootstrap.org/augdocs/friedewald030402/augmentinghumanintellect/ahi62index.html
9. Engelbart, Douglas, "Draft OHS-Project Plan," October 23, 2000. On the web at http://www.bootstrap.org/augdocs/bi-2120.html
10. Gradman, Eric, "Distributed Social Software," December 2003. On the web at http://www.gradman.com/projects/dss/final/final.pdf
11. Huang, Yifen; Govindaraju, Dinesh; Mitchell, Tom; Rocha de Carvalho, Vitor; Cohen, William.. "Inferring Ongoing Activities of Workstation Users by Clustering Email." In Proceedings of CEAS, First Conference on Email and Anti-Spam (CEAS). July 2004. On the web at http://www.ceas.cc/papers-2004/149.pdf
12. Karger, David R., Karun Bakshi, David Huynh, Dennis Quan, and Vineet Sinha, "Haystack: A Customizable General-Purpose Information Management Tool for End Users of Semistructured Data," in CIDR 2005, Asilomar, California. On the web at http://www-db.cs.wisc.edu/cidr/cidr2005/papers/P02.pdf
13. McCluhan, Marshal, *The Medium is the Message*, Wired Books, 1996.
14. Mill, John Stuart, *Autobiography*, 1873. On the web at http://www.utilitarianism.com/millauto/seven.html
15. Nelson, Theodor von Holm. "Xanalogical Structure, Needed Now More than Ever: Parallel Documents, Deep Links to Content, Deep Versioning, and Deep Re-Use." ACM Computing Surveys 31(4), December 1999. On the web at http://www.cs.brown.edu/memex/ACM_HypertextTestbed/papers/60.html
16. Osinski, Stanislaw; Stefanowski, Jerzy; Weiss, Dawid. "Lingo: Search Results Clustering Algorithm Based on Singular Value Decomposition." Intelligent Information Systems 2004: 359-368. On the web at http://www.cs.put.poznan.pl/dweiss/site/publications/download/iipwm-osinski-weiss-stefanowski-2004-lingo.pdf
17. Park, Jack, Editor, and Sam Hunting, Technical Editor, *XML Topic Maps: Creating and Using Topic Maps for the Web*, Boston, MA. Addison-Wesley, 2003

18. Sauermann, Leo, “The Gnowsis Semantic Desktop for Information Integration,” in IOA 2005, Kaiserlauten, Germany. On the web at http://www.dfki.uni-kl.de/~sauermann/papers/Sauermann2005a.pdf
19. Thomas, Dave, “Open Augment – Back To The Future Preserving the Augment Legacy With XML,” XML 2003 Conference. Philadelphia, PA. December 2003. On the web at http://www.idealliance.org/papers/dx_xml03/papers/05-00-00/05-00-00.pdf